DEVIEW 2016

딥러닝과 강화 학습으로 나보다 잘하는 쿠키런 AI 구현하기

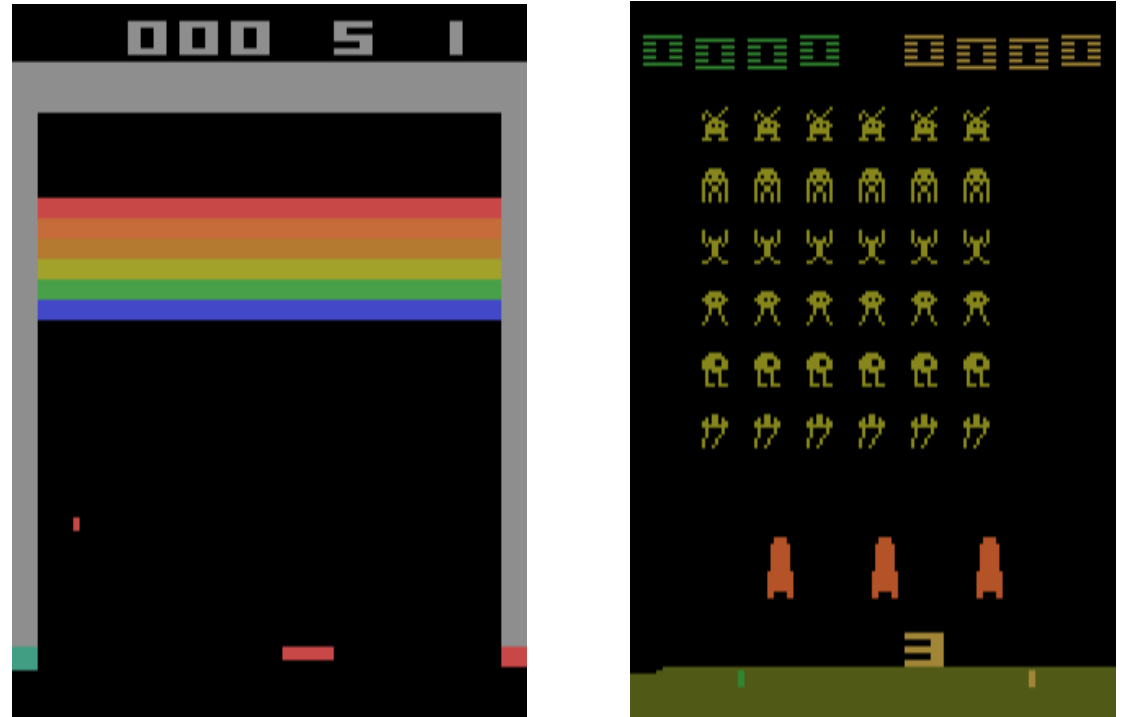
김태훈
DEVSISTERS

저는

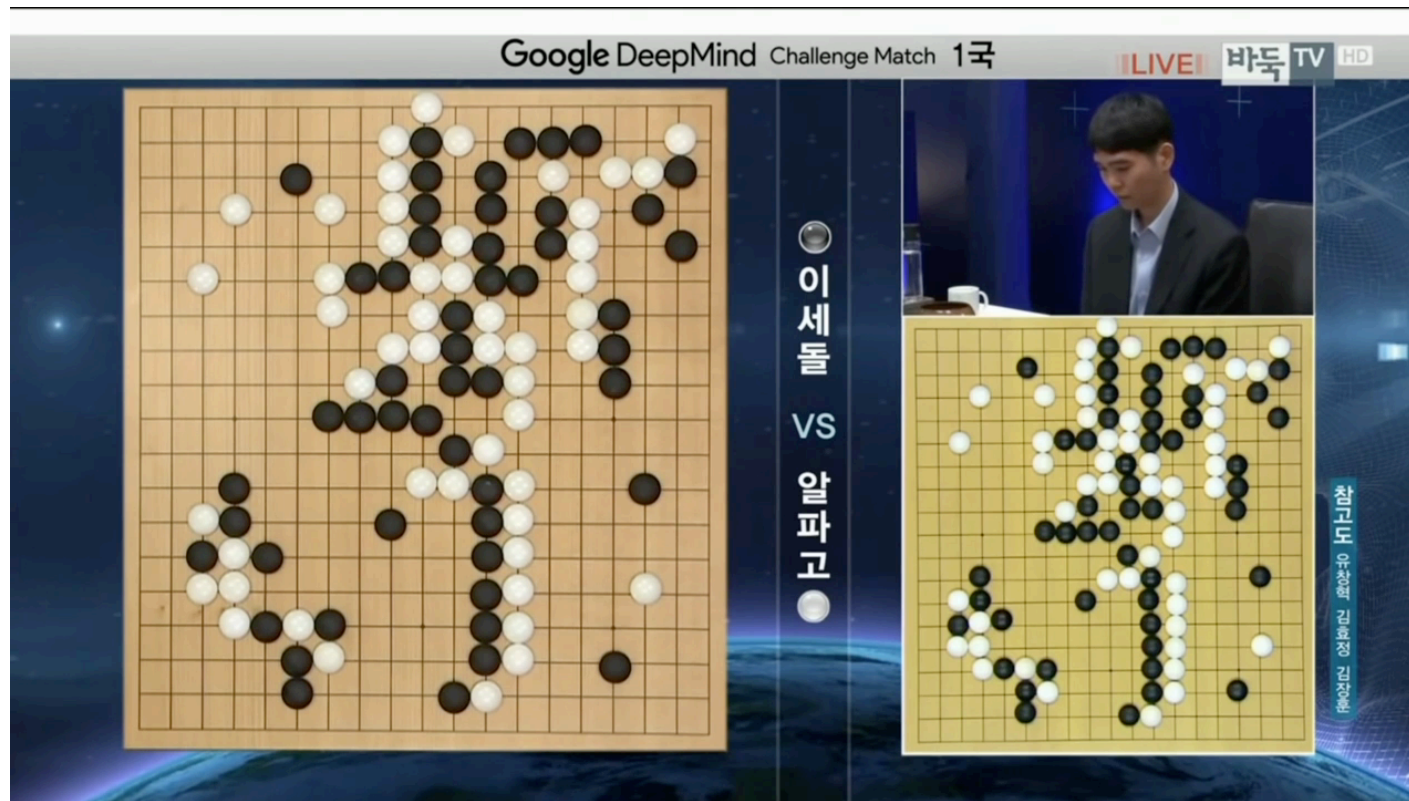
UNIST 졸업

DEVSISTERS 병특

EMNLP, DataCom, IJBDI 논문 게재

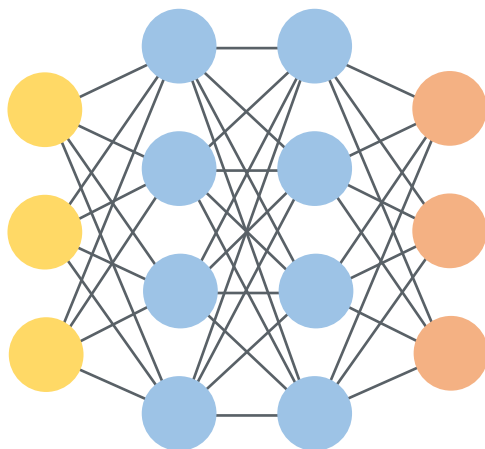
http://carpedm20.github.io

딥러닝 + 강화 학습

Playing Atari with Deep Reinforcement Learning (2013)

https://deepmind.com/research/dqn/

Mastering the game of Go with deep neural networks and tree search (2016)

https://deepmind.com/research/alphago/

VIZDOOM (2016)

http://vizdoom.cs.put.edu.pl/

딥러닝 + 강화 학습

딥러닝 + 강화 학습

“뉴럴뉴럴”한 뉴럴 네트워크

딥러닝 + 강화 학습?

Reinforcement Learning

Machine Learning

지도 학습

지도 학습
비지도 학습

지도 학습
비지도 학습
강화 학습

지도 학습
비지도 학습
강화 학습

지도 학습

동전

음식

동전

음식

지도 학습

지도 학습

분류

Classification

지도 학습
비지도 학습
강화 학습

비지도 학습

비지도 학습

비지도 학습

군집화

Clustering

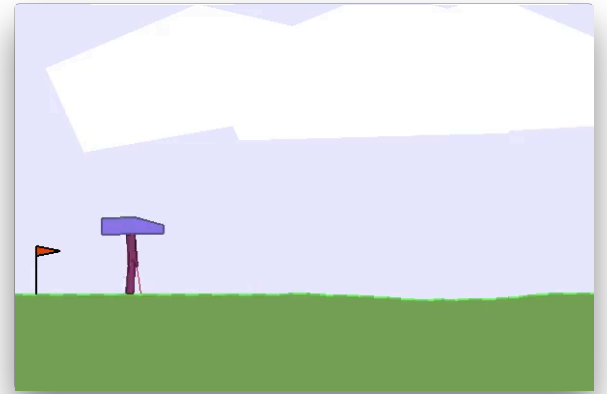
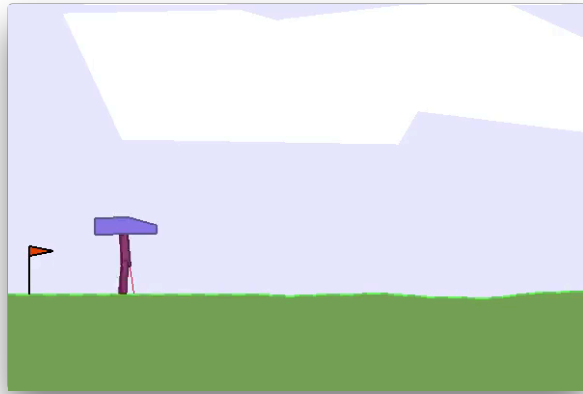
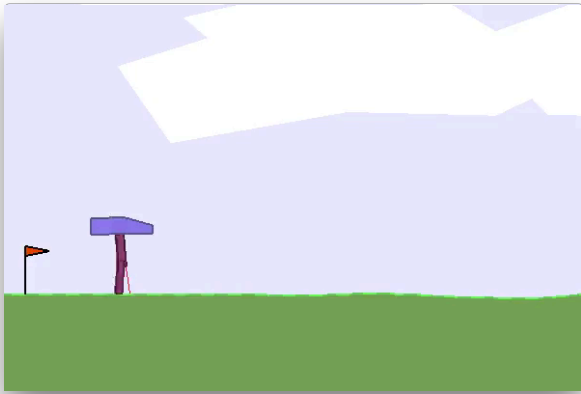
지도 학습
비지도 학습
강화 학습

분류도 아니고 군집화도 아닌것?

로봇를 걷게 하려면?

처음에는 학습할 데이터가 없다

조금씩 관절을 움직여 보면서

(처음에는 아무것도 모르니 랜덤으로)

시행 착오로부터 배운다

(정답이 없기 때문에 학습 결과는 다양함)

강화 학습

(Reinforcement Learning)

목표

목표

목표

목표

Agent

Agent

Environment

Agent
State s_t
Environment

Agent
State s_t
Environment
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 1 1 3 3 0 0 0
0 1 1 3 3 0 0 0
0 1 1 0 0 0 0 0
-1 -1 -1 -1 -1 -1 -1 -1

Agent
State s_t
Action $a_t = 2$
Environment
점프

Agent
State s_t
Action $a_t = 2$
Environment
점프

Agent

State s_t

Reward $r_t = 1$

Action $a_t = 2$

Environment

Agent
State s_t
Reward $r_t = 1$
Action $a_t = 2$
Environment

Agent

State s_t

Environment

Agent
Environment
State s_t
가만히
Action $a_t = 0$

Agent
State s_t
Reward $r_t = -1$
Action $a_t = 0$
Environment

즉, 강화 학습은

- Agent가 action을 결정하는 방법을 학습시키는 것
- 각 action은 그 다음 state에 영향을 끼친다
- 성공한 정도는 reward로 측정
- 목표 : 미래의 reward가 최대가 되도록 action을 취하는 것

Reinforcement Learning

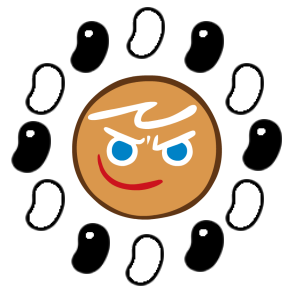
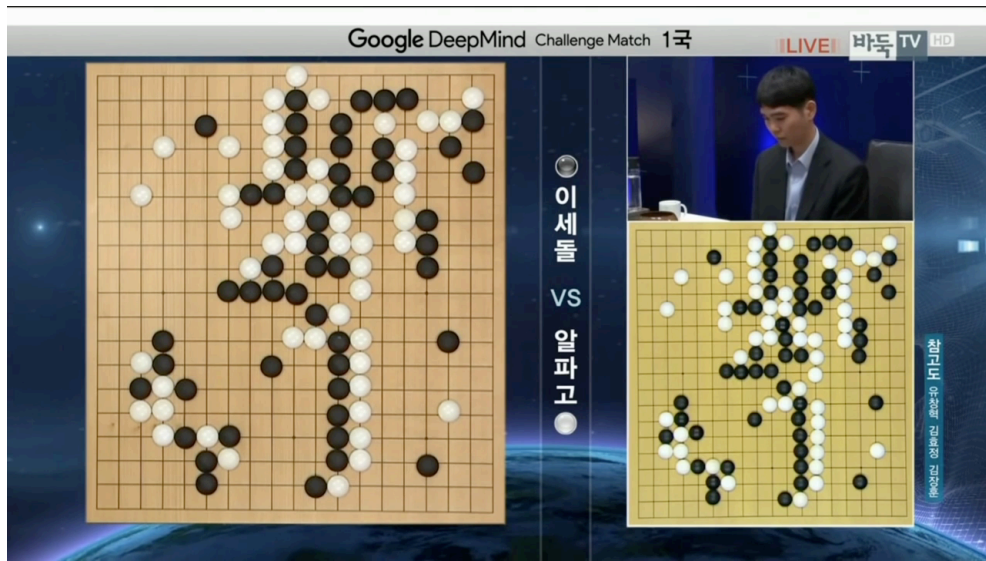
그래서,

DL+RL 로 무엇을 했나?

AlphaRun

Google DeepMind Challenge Match 1국
LIVE
바둑TV
이세돌
VS
알파고

+

쿠키런
오븐브레이크

쿠키가 스스로 달릴 수 있으면?

게임 밸런싱을

자동으로 할 수 있지 않을까?

쿠키 30개 (평균 8레벨)

펫 30개

보물 9개 (2개씩 장착)

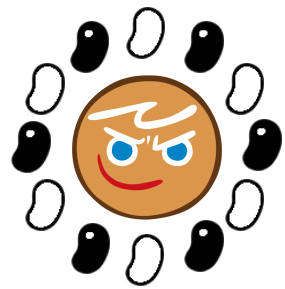
맵 7개

평균 플레이 4분

$30\times8\times30\times{}_9C_2\times7\times4=$

5,040일

평균 플레이 4초

1대 × 6개 프로세스

14일

AlphaRun

쿠키런 A.I.를 지향하는 기술들

DEVSISTERS

쿠키런 AI를 지탱하는 8가지 기술

1. Deep Q-Network (2013)
2. Double Q-Learning (2015)
3. Dueling Network (2015)
4. Prioritized Experience Replay (2015)
5. Model-free Episodic Control (2016)
6. Asynchronous Advantageous Actor-Critic method (2016)
7. Human Checkpoint Replay (2016)
8. Gradient Descent with Restart (2016)

쿠키런 AI를 지탱하는 8가지 기술

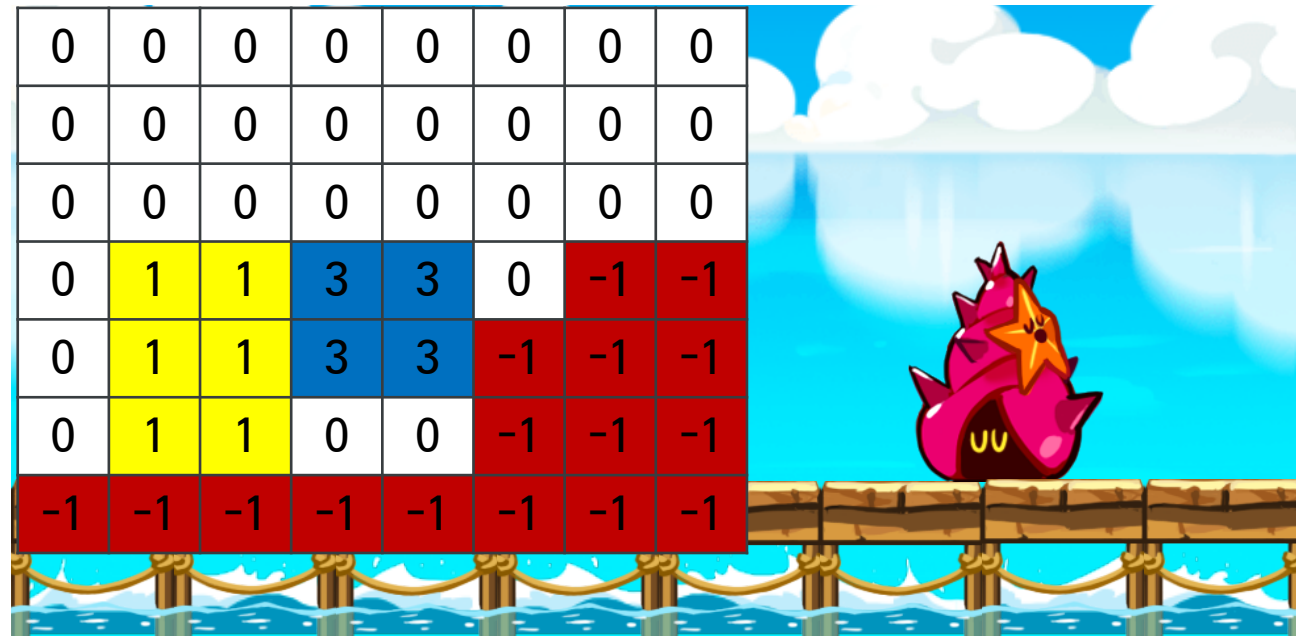
1. Deep Q-Network (2013)
2. Double Q-Learning (2015)
3. Dueling Network (2015)
4. Prioritized Experience Replay (2015)
5. Model-free Episodic Control (2016)
6. Asynchronous Advantageous Actor-Critic method (2016)
7. Human Checkpoint Replay (2016)
8. Gradient Descent with Restart (2016)

1. Deep Q-Network

State s_t

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	1	1	3	3	0	-1	-1
0	1	1	3	3	-1	-1	-1
0	1	1	0	0	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1

Action $a_t = ?$

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	1	1	3	3	0	-1	-1
0	1	1	3	3	-1	-1	-1
0	1	1	0	0	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1

Action a_t

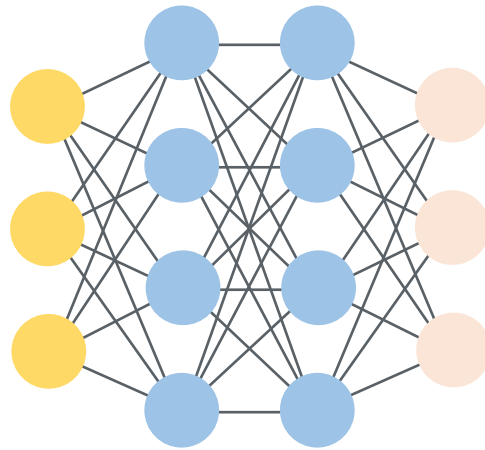
? ? ?

0

None 슬라이드 점프

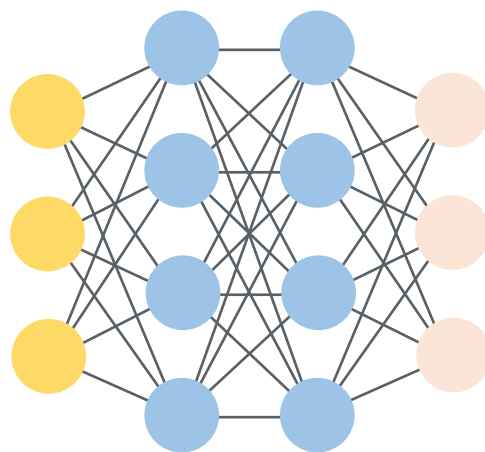
Action $a_t = ?$

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	1	1	3	3	0	-1	-1
0	1	1	3	3	-1	-1	-1
0	1	1	0	0	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1

s_t

가장 좋은 행동

Action $a_t = ?$

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	1	1	3	3	0	-1	-1
0	1	1	3	3	-1	-1	-1
0	1	1	0	0	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1

s_t

$$Q(s, a)$$

State s 에서

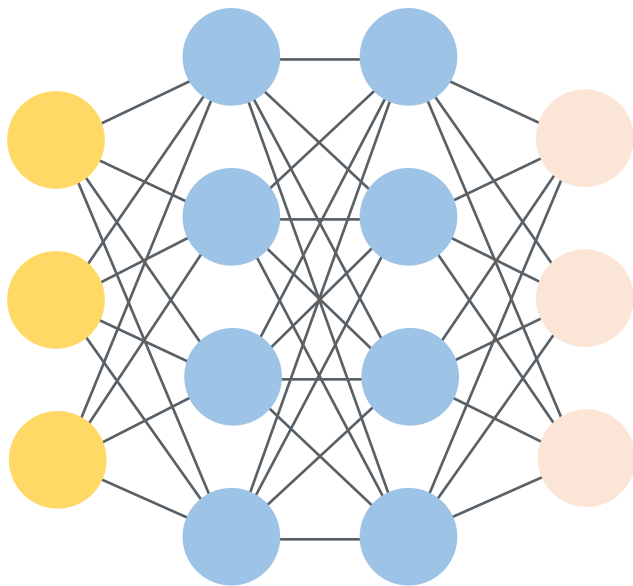
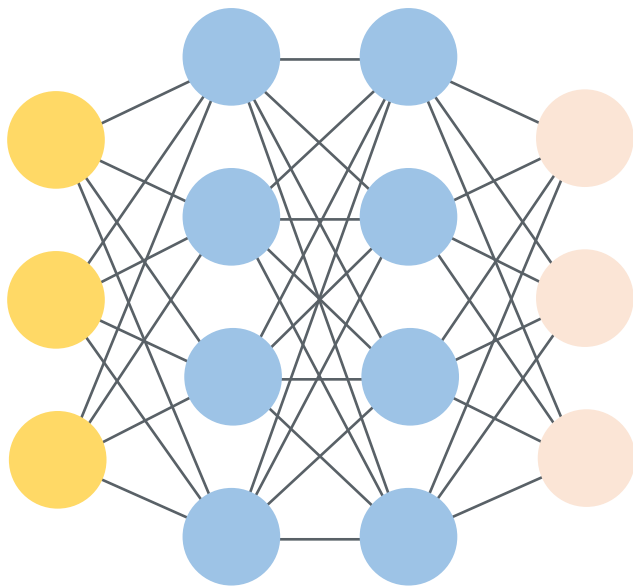
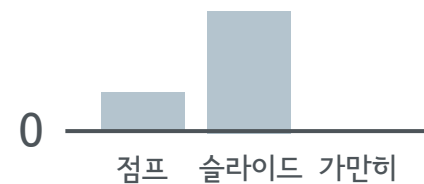
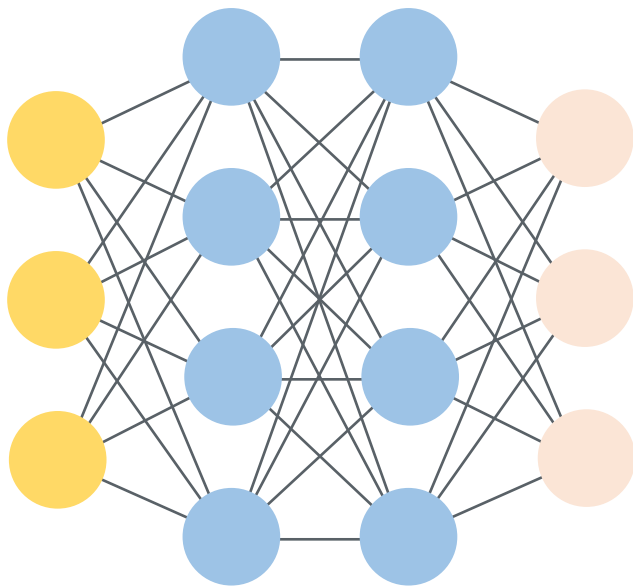
Action a 를 했을 때

기대되는 미래 가치 Q

s
Q(s, 점프)
Q(s, 슬라이드)
Q(s, 가만히)

0
점프
슬라이드
가만히

s
Q(s, 점프) = 1
Q(s, 슬라이드) = 5
Q(s, 가만히) = 0

0
점프
슬라이드
가만히

S
Q(s, 점프) = 1
Q(s, 슬라이드) = 5
Q(s, 가만히) = 0

Q : 기대되는 미래 가치

쿠키런 오븐브레이크 ’s 가치 =

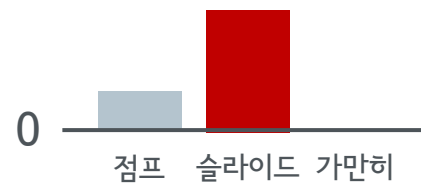
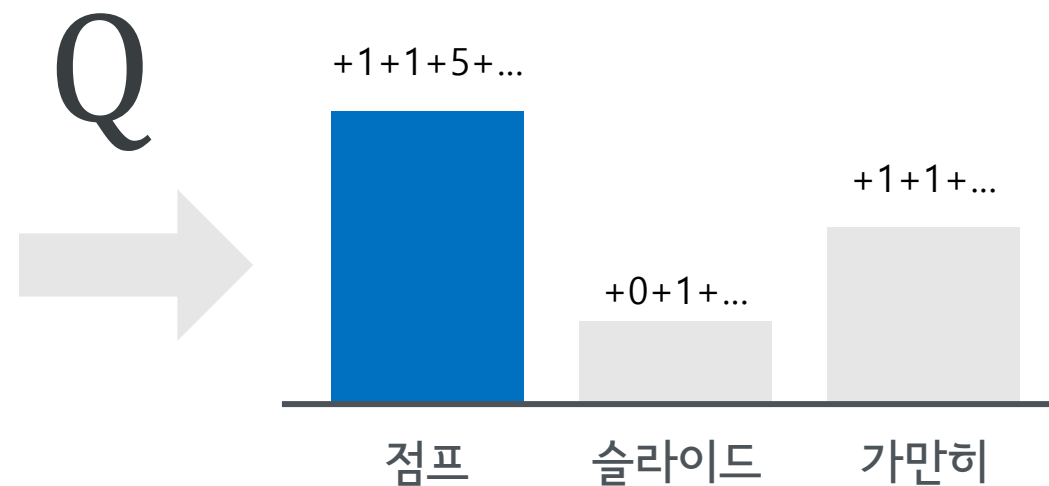
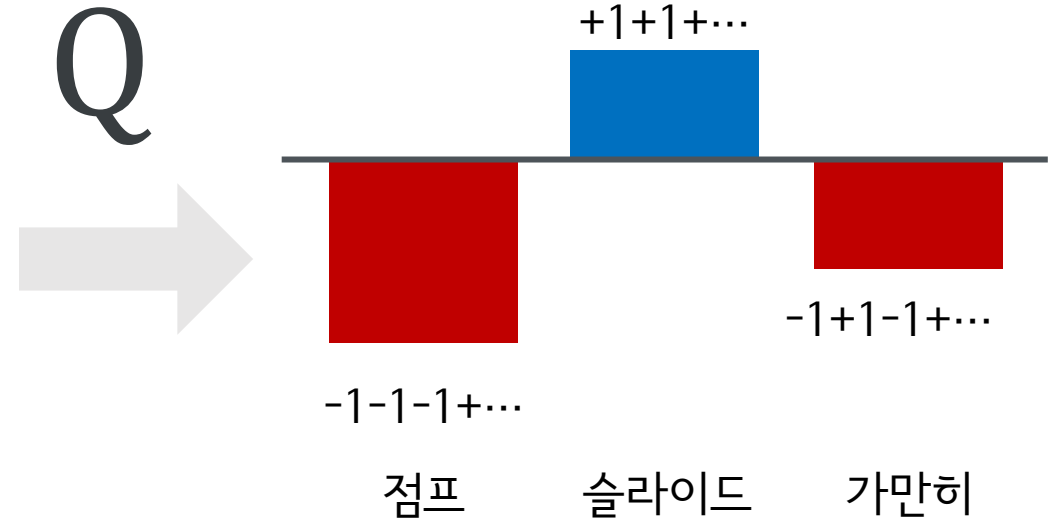
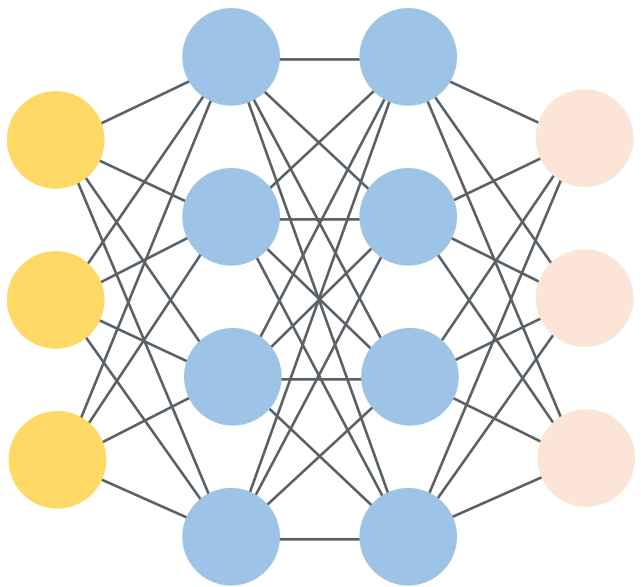
쿠키런 오븐브레이크 's 가치 = 점수

Q

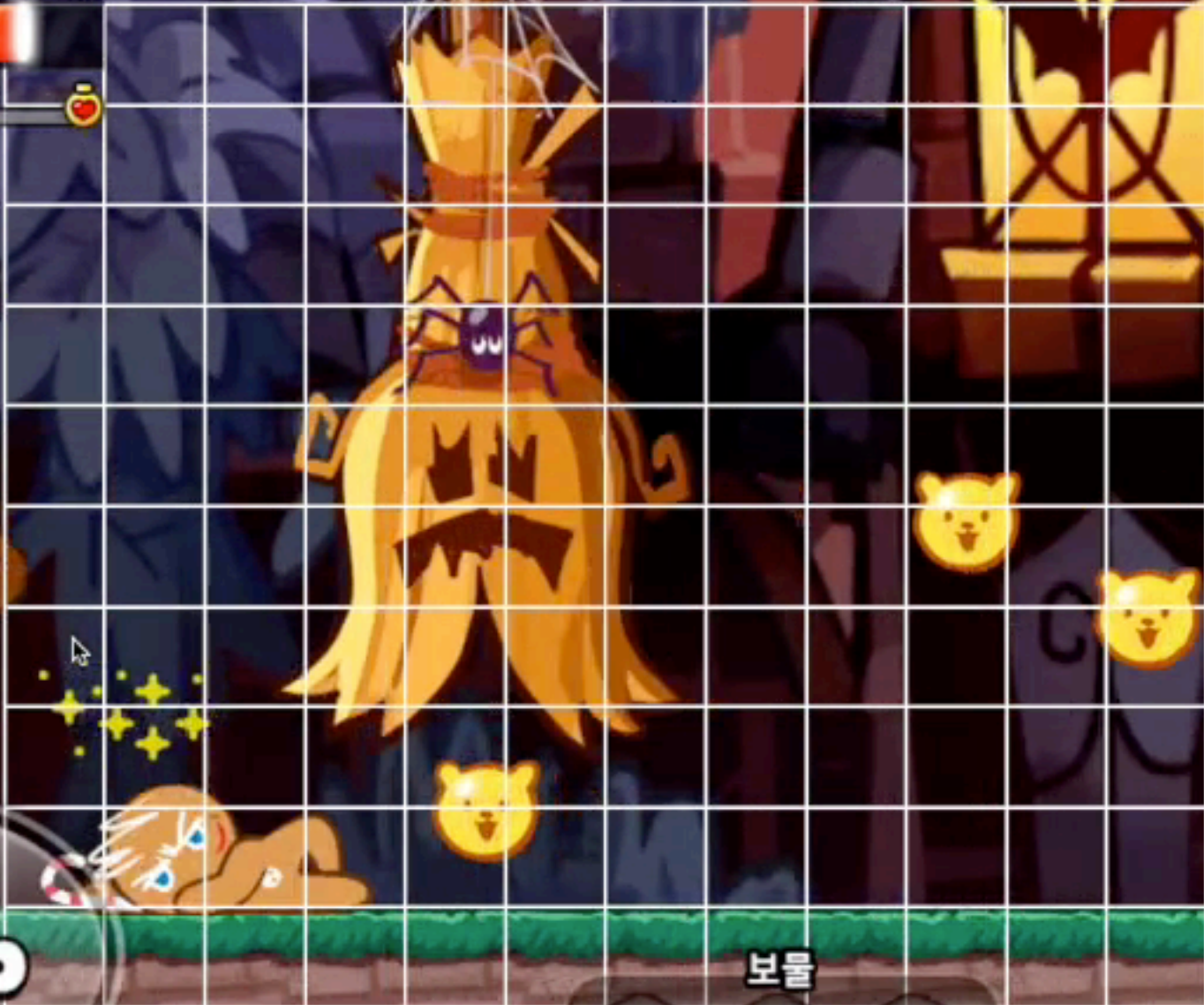
미래에 얻을 점수들의 합

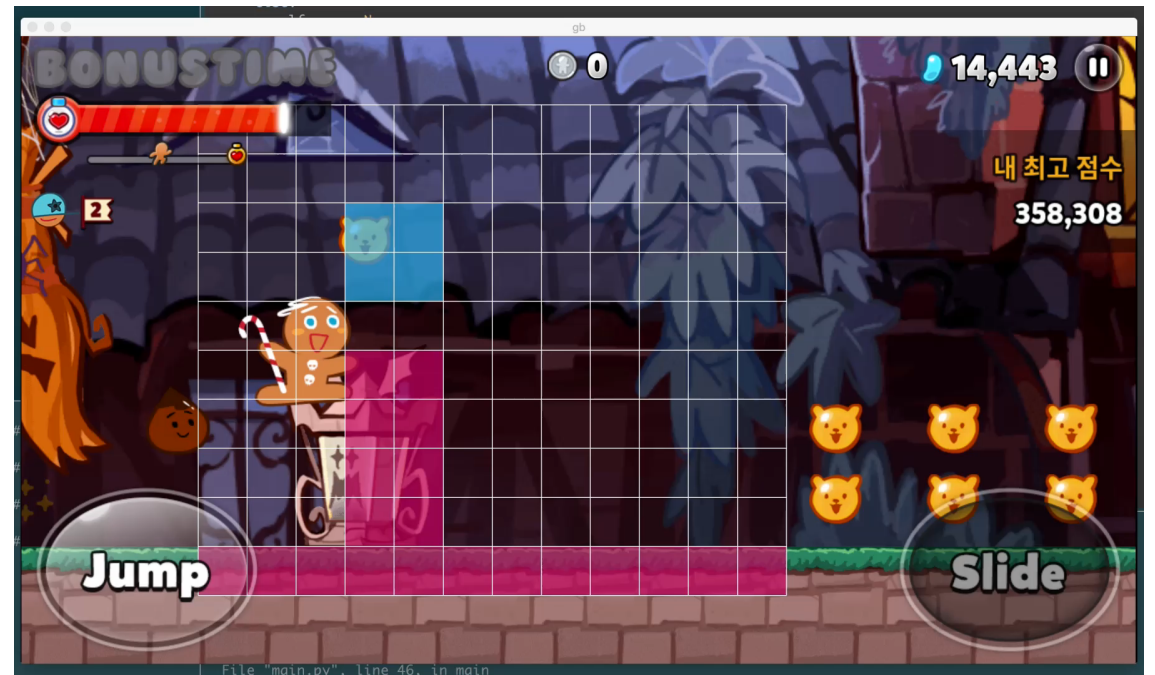
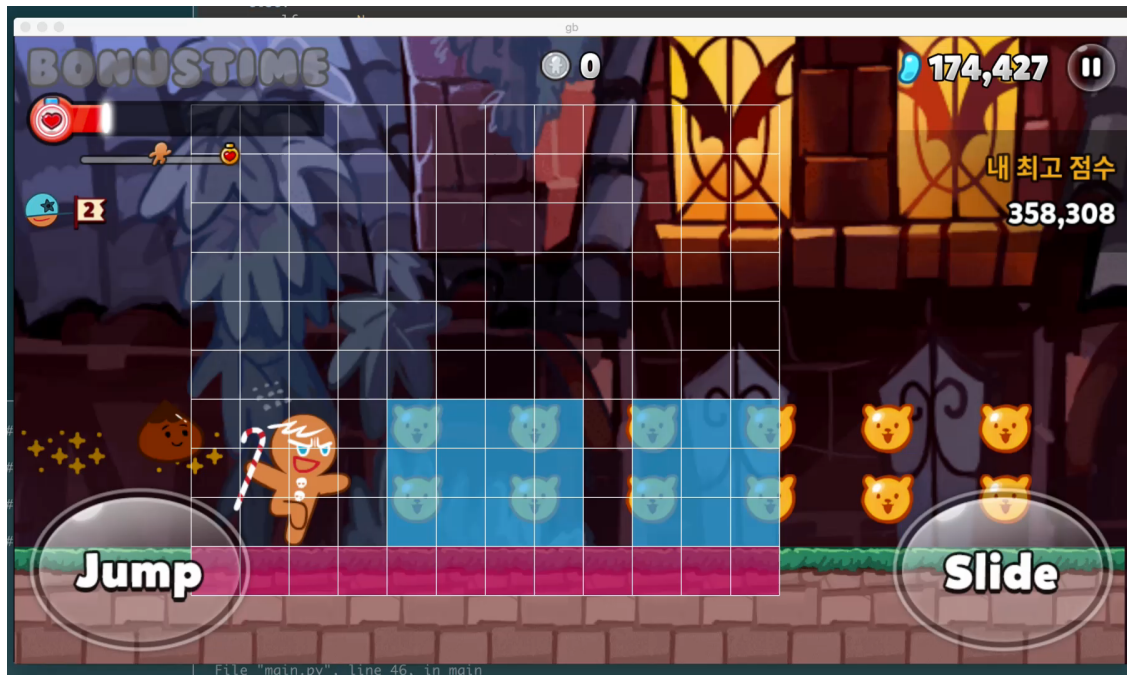
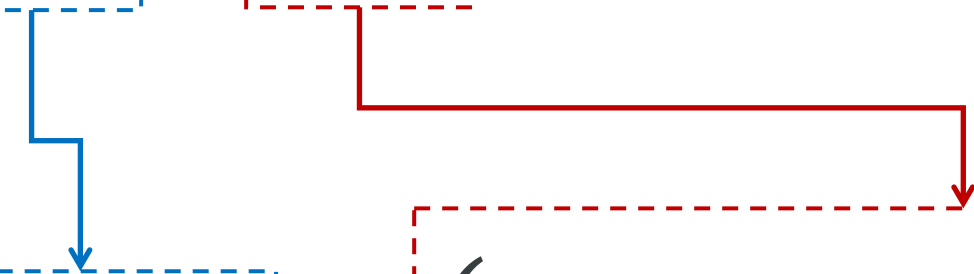
BONUSTIME
278
78/400
1
1,122,876
Jump
Slide
GL verts: 1166
GL calls: 24
60.5 / 0.014

Q
+1+1+5+...
+0+1+...
+1+1+...
점프
슬라이드
가만히

BONUSTIME
0
24,442
내 최고 점수
369,815
Jump
Slide
보물

Q
+1+1+…
-1-1-1+…
-1+1-1+…
점프
슬라이드
가만히

$$loss = \left(r + \gamma \max_{a`} \hat{Q}(s, a`) - Q(s, a) \right)^2$$

결과는?

BONUSTIME
0
48,884
2
내 최고 점수
369,815
Jump
Slide
보물

하지만 #1...

- 왜 하나씩 놓치고, 이상한 행동을 하는 걸까

2. Double Q-Learning

핵심은,

미래의 가치를 나타내는 Q가

낙관적 예측 or **발산**하는 것을 **막음**

$$loss = \left(Q(s,a) - \left(\mathrm{r} + \gamma \max_{a`} \hat{Q}(s,a`) \right) \right)^2$$

$$loss = (\text{예측} - \text{정답})^2$$

$$loss = \left(Q(s,a) - \left(\mathrm{r} + \gamma \max_{a`} \hat{Q}(s, a`) \right) \right)^2$$

예측 – 정답

$$loss = \left(Q(s,a) - \left(r + \gamma \max_{a`} \hat{Q}(s,a`) \right) \right)^2$$

0

예측 – 정답

$$loss = \left(Q(s,a) - \left(\mathrm{r} + \gamma \max_{a`} \hat{Q}(s, a`) \right) \right)^2$$

0

예측 – 정답

$$loss = \left(Q(s,a) - \left(\mathrm{r} + \gamma \max_{a`} \hat{Q}(s,a`)\right)\right)^2$$

0
예측 – 정답
$loss = \left(Q(s,a) - \left(r + \gamma \max_{a`} \hat{Q}(s,a`) \right) \right)^2$

0
Q
예측 – 정답
$loss = \left(Q(s,a) - \left(r + \gamma \max_{a`} \hat{Q}(s,a`) \right) \right)^2$

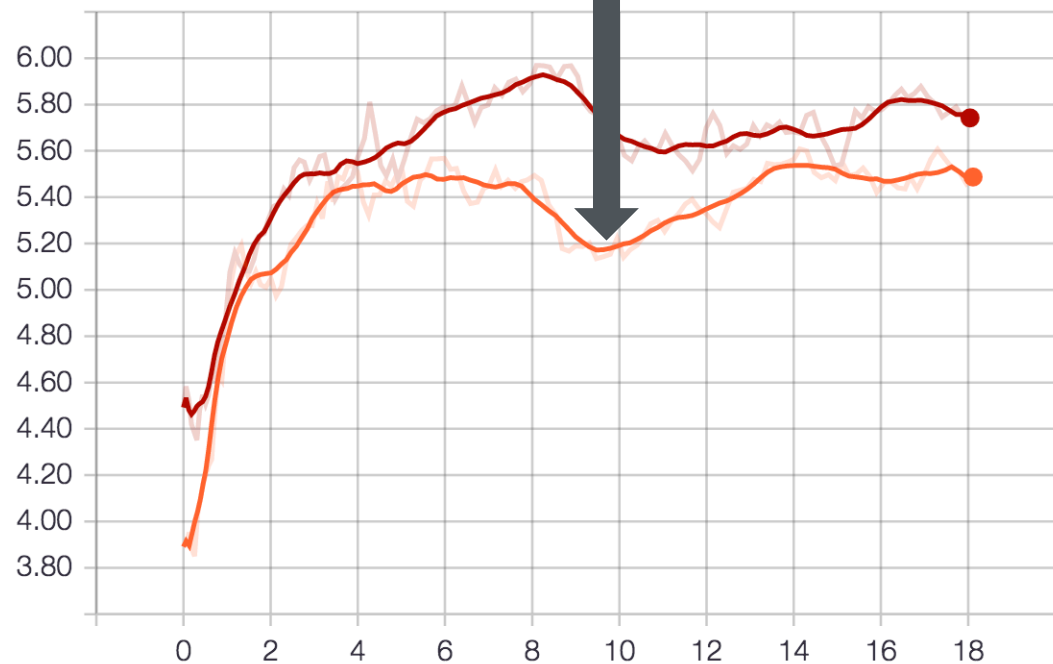
DQN

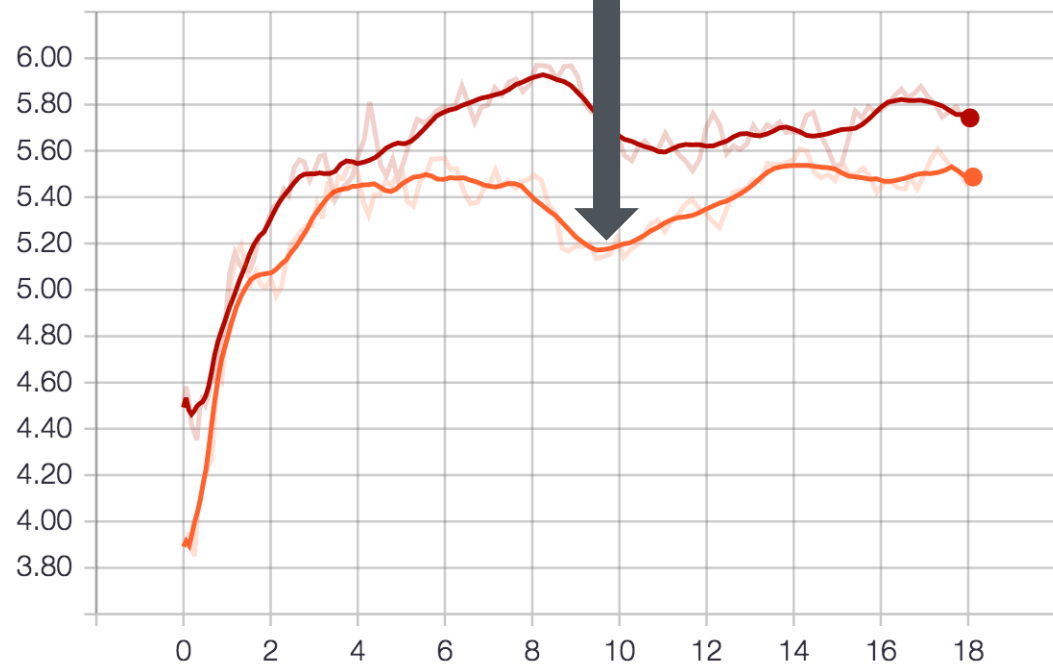
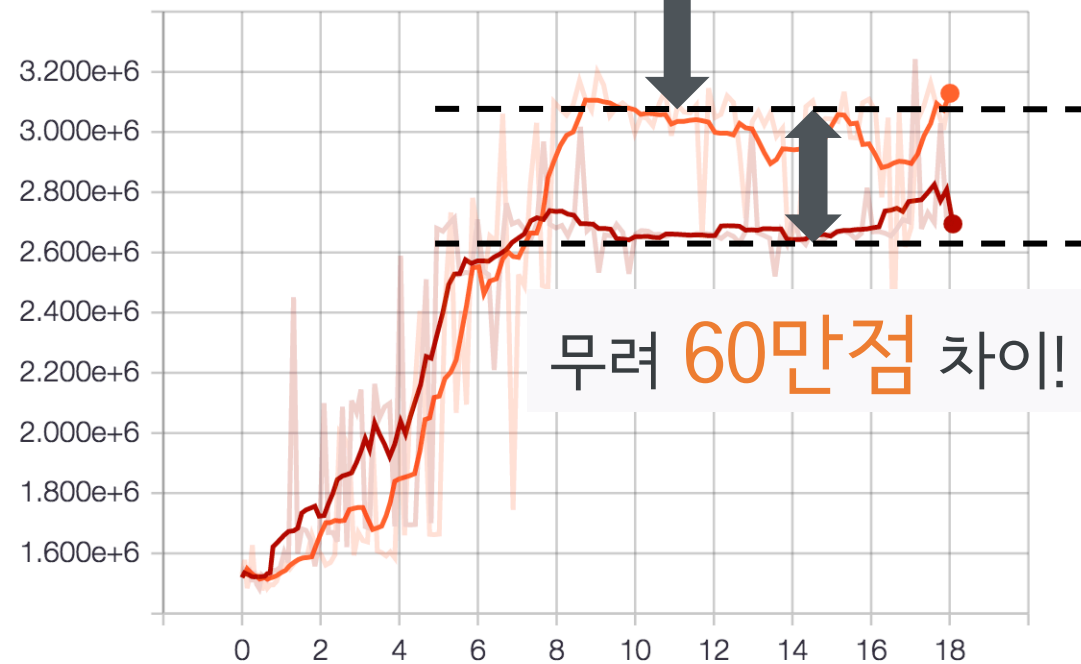
$$loss = \left(\mathrm{r} + \gamma \max_{a`} \hat{Q}(s, a`) - Q(s,a) \right)^2$$

$$= \left(\mathrm{r} + \gamma \hat{Q}\left(s, \arg\max_{a`} \hat{Q}(s, a`)\right) - Q(s,a) \right)^2$$

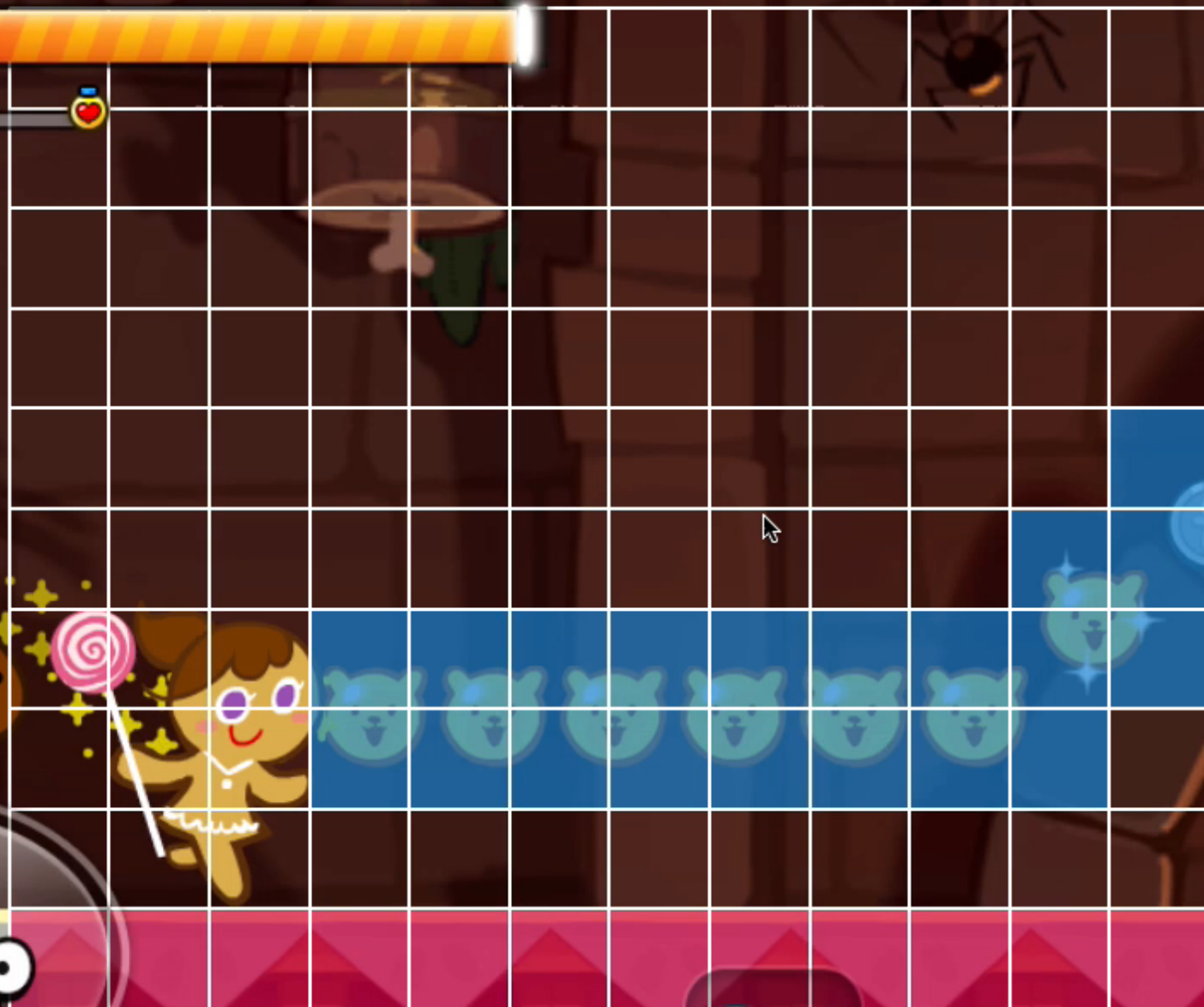
Double DQN

$$loss = \left(\mathrm{r} + \gamma \hat{Q}\left(s, \arg\max_{a`} Q(s, a`)\right) - Q(s,a) \right)^2$$

Double Q가 Q 값은 작지만

— : Deep Q-learning — : Double Q-learning

Double Q가 Q 값은 작지만

점수는 훨씬 높다!

— : Deep Q-learning — : Double Q-learning

결과는?

BONUSTIME
341
77,414
10/10
내 최고 점수
Jump
Slide
verts: 1262
calls: 33

“아, 다했다.”

하지만 #2...

- 단조로운 land 1이 아닌 land 3를 가보니...

하지만 #2...

- 그리고 충격과 공포의 **보너스 타임**...

3. Dueling Network

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	1	1	3	3	0	0	0
0	1	1	3	3	0	0	0
0	1	1	0	0	0	0	0
-1	-1	-1	-1	-1	-1	-1	-1

기대되는 **미래 가치**

$Q(s, a)$의 값은?

+1+1+1…

앞에 젤리가 많은지,

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	1	1	3	3	0	0	0
0	1	1	3	3	0	0	0
0	1	1	0	0	0	0	0
-1	-1	-1	-1	-1	-1	-1	-1

0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
3	3	3	3	0	0
3	3	3	3	0	0
0	0	0	0	0	0
-1	-1	-1	-1	-1	-1

+1+1+1…

+0-1-1+…

앞에 젤리가 많은지, 장애물이 많은지 전혀 알 수 없음

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	1	1	3	3	0	0	0
0	1	1	3	3	0	0	0
0	1	1	0	0	0	0	0
-1	-1	-1	-1	-1	-1	-1	-1

0	0	0	0	0	0
0	0	0	-1	-1	0
0	0	0	-1	-1	0
-1	-1	0	-1	-1	0
-1	-1	0	-1	-1	0
-1	-1	0	-1	-1	0
-1	-1	-1	-1	-1	-1

정확한 Q 예측이 어렵다

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	1	1	3	3	0	0	0
0	1	1	3	3	0	0	0
0	1	1	0	0	0	0	0
-1	-1	-1	-1	-1	-1	-1	-1

점프 : 10? 8?

슬라이드 : -2? 1?

가만히 : 5? 12?

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	1	1	3	3	0	0	0
0	1	1	3	3	0	0	0
0	1	1	0	0	0	0	0
-1	-1	-1	-1	-1	-1	-1	-1

하지만!

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	1	1	3	3	0	0	0
0	1	1	3	3	0	0	0
0	1	1	0	0	0	0	0
-1	-1	-1	-1	-1	-1	-1	-1

Q를 정확하게 **예측**할 **필요**가 있을까?

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	1	1	3	3	0	0	0
0	1	1	3	3	0	0	0
0	1	1	0	0	0	0	0
-1	-1	-1	-1	-1	-1	-1	-1

슬라이드 : x (기준)

점프 : $x+1$? $x+3$?

가만히 : $x+1$? $x+2$?

10? 20?

0? 1?

14? 32?

0 (기준)

+1? +3?

-1? -2?

어느 것이 **예측**하기 더 **쉬울까**?

당연히 차이를 배우는 것이 쉽다

$Q(s,a)$

Value

$$Q(s,a) = V(s)$$

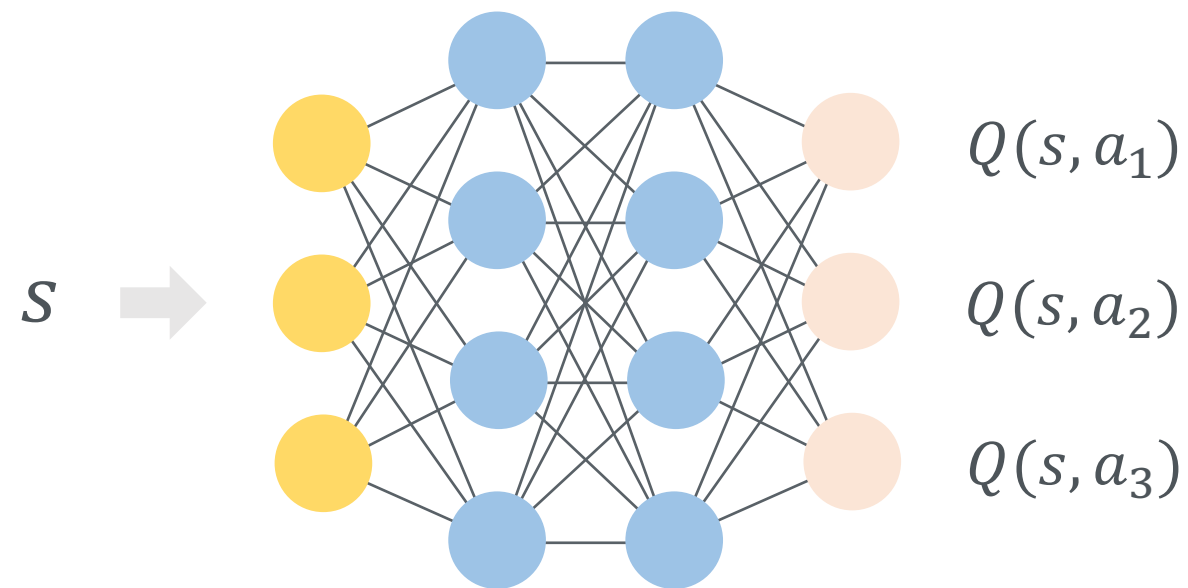
기준점 x

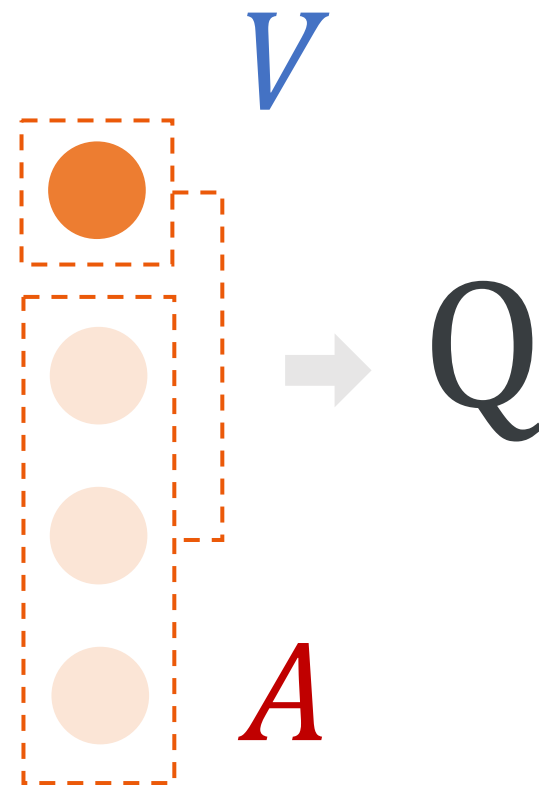
Value

Advantage

$$Q(s,a) = V(s) + A(s,a)$$

기준점 x

상대적인 Q값의 차이

Deep Q-Network

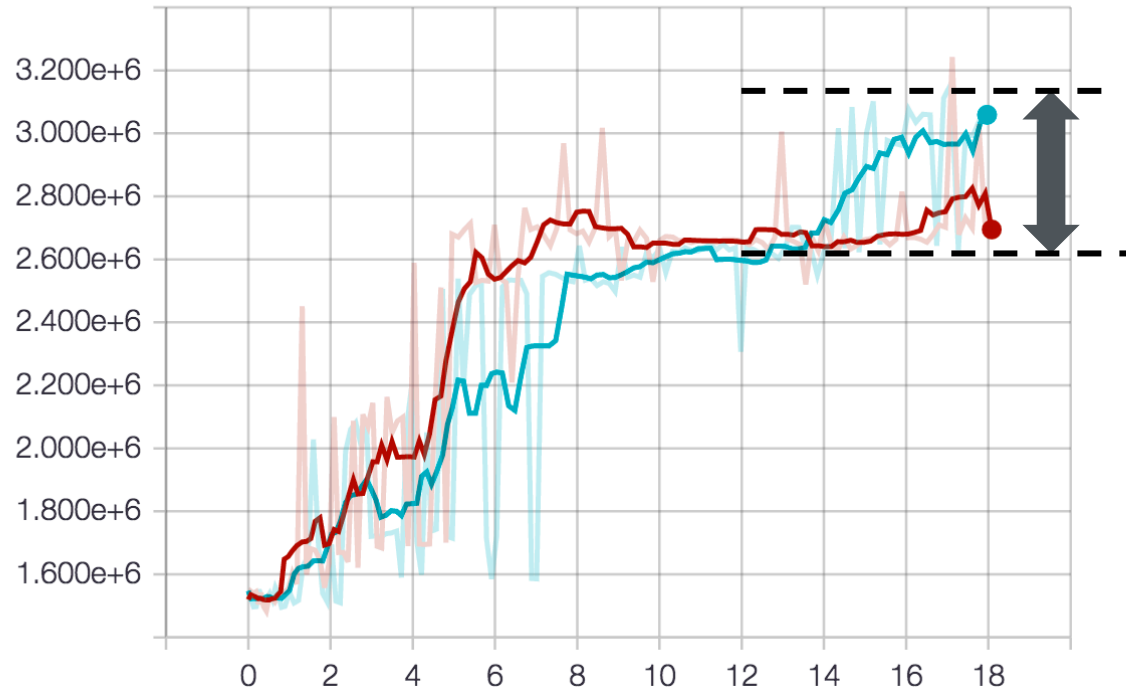
Dueling Network

Dueling Network

V
Q
A

Sum : $Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + A(s,a;\theta,\alpha)$

Max : $Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + \left(A(s,a;\theta,\alpha) - \max_{a` \in \mathcal{A}} A(s,a`;\theta,\alpha)\right)$

Average: $Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + \left(A(s,a;\theta,\alpha) - \frac{1}{|\mathcal{A}|}\sum_{a`} A(s,a`;\theta,\alpha)\right)$

3. Dueling network

— : DQN — : Sum — : Max

3. Dueling network

쿠키런 AI를 지탱하는 8가지 기술

1. Deep Q-Network (2013)
2. Double Q-Learning (2015)
3. Dueling Network (2015)
4. Prioritized Experience Replay (2015)
5. Model-free Episodic Control (2016)
6. Asynchronous Advantageous Actor-Critic method (2016)
7. Human Checkpoint Replay (2016)
8. Gradient Descent with Restart (2016)

쿠키런 AI를 지탱하는 8가지 기술

1. Deep Q-Network (2013)
2. Double Q-Learning (2015)
3. Dueling Network (2015)
4. Prioritized Experience Replay (2015)
5. Model-free Episodic Control (2016)
6. Asynchronous Advantageous Actor-Critic method (2016)
7. Human Checkpoint Replay (2016)
8. Gradient Descent with Restart (2016)

강화에 계속해서 실패한다면?

논문 8개나 갈아 넣었는데 안된다고..?

DEVSISTERS

엔지니어링~~이라고 쓰고 노가다라고 부른다~~

1. Hyperparameter tuning
2. Debugging
3. Pretrained model
4. Ensemble method

1. Hyperparameter tuning

네트워크 바꾸기
Optimizer 바꾸기
reward 식 바꾸기
...

총 70+개

Network

Environment

Experience memory

Training method

1. activation_fn : activation function (relu, tanh, elu, leaky)
2. initializer : weight initializer (xavier)
3. regularizer : weight regularizer (None, l1, l2)
4. apply_reg : layers where regularizer is applied
5. regularizer_scale : scale of regularizer
6. hidden_dims : dimension of network
7. kernel_size : kernel size of CNN network
8. stride_size : stride size of CNN network
9. dueling : whether to use dueling network
10. double_q : whether to use double Q-learning
11. use_batch_norm : whether to use batch normalization

1. screen_height : # of rows for a grid state
2. screen_height_to_use : actual # of rows for a state
3. screen_width : # of columns for a grid state
4. screen_width_to_us : actual # of columns for a state
5. screen_expr : actual # of row for a state
    - c : cookie
    - p : platform
    - s : score of cells
    - p : plain jelly
    - i : plain item
    - h : heal
    - m : magic
    - hi : height
    - sp : speed
    - rj : remained jump

    ex) (c+2*p)-s,i+h+m,[rj,sp], ((c+2*p)-s)/2.,i+h+m,[rj,sp,hi]
6. action_repeat : # of reapt of an action (frame skip)

1. history_length : the length of history
2. memory_size : size of experience memory
3. priority : whether to use prioritized experience memory
4. preload_memory : whether to preload a saved memory
5. preload_batch_size : batch size of preloaded memory
6. preload_prob : probability of sampling from pre-mem
7. alpha : alpha of prioritized experience memory
8. beta_start: beta of prioritized experience memory
9. beta_end_t: beta of prioritized experience memory

1. optimizer : type of optimizer (adam, adagrad, rmsprop)
2. batch_norm : whether to use batch normalization
3. max_step : maximum step to train
4. target_q_update_step : # of step to update target network
5. learn_start_step : # of step to begin a training
6. learning_rate_start : the maximum value of learning rate
7. learning_rate_end : the minimum value of learning rate
8. clip_grad : value for a max gradient
9. clip_delta : value for a max delta
10. clip_reward : value for a max reward
11. learning_rate_restart : whether to use learning rate restart

過猶不及

과유불급

성능은 올릴 수는 있지만,

그만큼 끊임없는 실험을 해야한다

고정시킬 변수들을 정해서

한동안 건드리지 않는다!

```
for land in range(3, 7):
  for cookie in cookies:
    options = []
    option = {
      'hidden_dims': ["[800, 400, 200]", "[1000, 800, 200]"],
      'double_q': [True, False],
      'start_land': land,
      'cookie': cookie,
    }
    options.append(option.copy())
    array_run(options, ’double_q_test')
```

실험, 실험, 실험, 실험, 실험,
실험, 실험, 실험, 실험, 실험,
실험, 실험, 실험, 실험, 실험,
실험, 실험, 실험, 실험, 실험,
실험, 실험, 실험, 실험, 실험,
실험, 실험, 실험, 실험, 실험,

2. Debugging

“쿠키가 이상하게 행동하는데

이유라도 알려줬으면...”

```
Previous REWARD: 0.08
             #              #              #              #
             #              #              #              #
*            # ***          #  ***         #    ***       #
*        @@@ # ***        @ #  ***         #    ***       #
*        @@@ # ***        @ #  ***         #    ***       #
*        @@@ # ***        @ #  ***         #    ***       #
*   @@@@@@   # ***  @@@@@@@ #  ***  @@@@@@ #  $$***  @@@@ #
*$$@@@@@@@   # *$$ @@@@@@@@ #  $$* @@@@@@@ #  $$*** @@@@@ #
 $$@    @@   #  $$@@@    @@ #  $$@@@@    @ #  $$@@@@@@    #
*$$********* # *$$********* # *$$********* # ************ #
ACT:            NOOP           SLIDE    JUMP
Q:         7.3161182       7.2896504       6.9471564
DIFF:      0.0264678       0.0000000      -0.3424940
V:         5.7694306
ADV:       1.5466874       1.5202198       1.1777259
```

History

```
Previous REWARD: 0.08
             #              #              #              #
             #              #              #              #
*            # ***          #  ***         #    ***       #
*        @@@ # ***        @ #  ***         #    ***       #
*        @@@ # ***        @ #  ***         #    ***       #
*        @@@ # ***        @ #  ***         #    ***       #
*   @@@@@@   # ***  @@@@@@@ #  ***  @@@@@@ #  $$***  @@@@ #
*$$@@@@@@@   # *$$ @@@@@@@@ #  $$* @@@@@@@ #  $$*** @@@@@ #
 $$@    @@   #  $$@@@    @@ #  $$@@@@    @ #  $$@@@@@@    #
*$$********* # *$$********* # *$$********* # ************ #
ACT:             NOOP            SLIDE    JUMP
Q:          7.3161182       7.2896504      6.9471564
DIFF:       0.0264678       0.0000000     -0.3424940
V:          5.7694306
ADV:        1.5466874       1.5202198      1.1777259
```

TIME
226,833

Q-value

```
Previous REWARD: 0.08
             #              #              #              #
             #              #              #              #
*            # ***          #  ***         #    ***       #
*        @@@ # ***        @ #  ***         #    ***       #
*        @@@ # ***        @ #  ***         #    ***       #
*        @@@ # ***        @ #  ***         #    ***       #
*   @@@@@@   # ***  @@@@@@@ #  ***  @@@@@@ #  $$***  @@@@ #
*$$@@@@@@@   # *$$ @@@@@@@@ #  $$* @@@@@@@ #  $$*** @@@@@ #
 $$@    @@   #  $$@@@    @@ #  $$@@@@    @ #  $$@@@@@@    #
*$$********* # *$$********* # *$$********* # ************ #
ACT:            NOOP           SLIDE    JUMP
Q:         7.3161182       7.2896504       6.9471564
DIFF:      0.0264678       0.0000000      -0.3424940
V:         5.7694306
ADV:       1.5466874       1.5202198       1.1777259
```

ACT:	NOOP	SLIDE	JUMP
Q:	7.3161182	7.2896504	6.9471564
DIFF:	0.0264678	0.0000000	-0.3424940
V:	5.7694306		
ADV:	1.5466874	1.5202198	1.1777259

Dueling Network

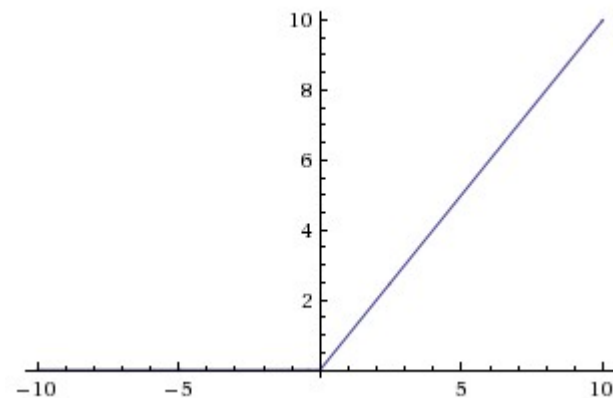
V(s)

A(s,a)

도움이 된 순간

gb
BONUSTIME
0
14,443
내 최고 점수
358,308
2
Jump
Slide
File "main.py", line 46, in main

도움이 된 순간

- 모든 action에 대해서 Q 값이 0
- State 값을 조금 바꿔서 출력해도 여전히 0
- Tensorflow의 `fully_connected` 함수에 activation function의 default값이 `nn.relu`로 설정되어 있음
- Activation function을 None으로 지정하니 해결!

```
146  -                               scope='value_hid')
147  -        value = fully_connected(value_hid, 1,
148  -                            weights_initializer=initializer,
149  -                            weights_regularizer=weights_regularizer,
150  -                            scope='value')
151  -
152  -        advantage_hid = fully_connected(layer, hidden_dim,
153  -                                activation_fn=activation_fn,
```

```
146  +                               scope='value_hid')
147  +        self.value = fully_connected(self.value_hid, 1,
148  +                            activation_fn=None,
149  +                            weights_initializer=initializer,
150  +                            weights_regularizer=weights_regularizer,
151  +                            scope='value')
152  +
153  +        self.advantage_hid = fully_connected(layer, hidden_dim,
```

tensorflow/contrib/layers/python/layers/layers.py

```
727  @add_arg_scope
728  def fully_connected(inputs,
729                      num_outputs,
730                      activation_fn=nn.relu,
731                      normalizer_fn=None,
732                      normalizer_params=None,
```

```
755    Args:
756      inputs: A tensor of with at least rank 2 and value fo
757        i.e. `[batch_size, depth]`, `[None, None, None, cha
758      num_outputs: Integer, the number of output units in t
759      activation_fn: activation function.
760      normalizer_fn: normalization function to use instead
```

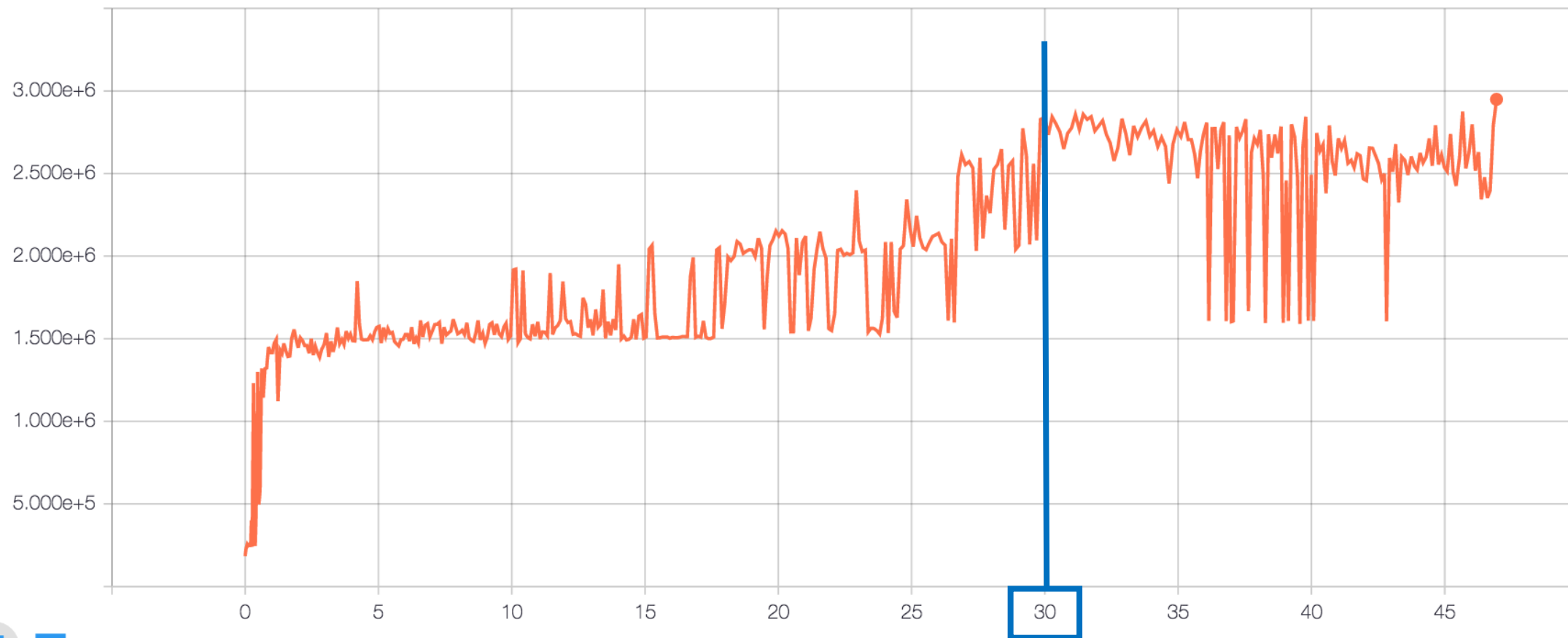
시도해 봤다면 좋았을 디버깅

- State를 실시간으로 수정하면서 Q 변화 보기

  ex) 젤리를 쿠키 앞에 그려보면서 변화 확인

- Exploration을 어떻게 하고 있는지
  - reward는 제대로 학습 될 수 있도록 정해져 있었는지?

3. Pretrained model

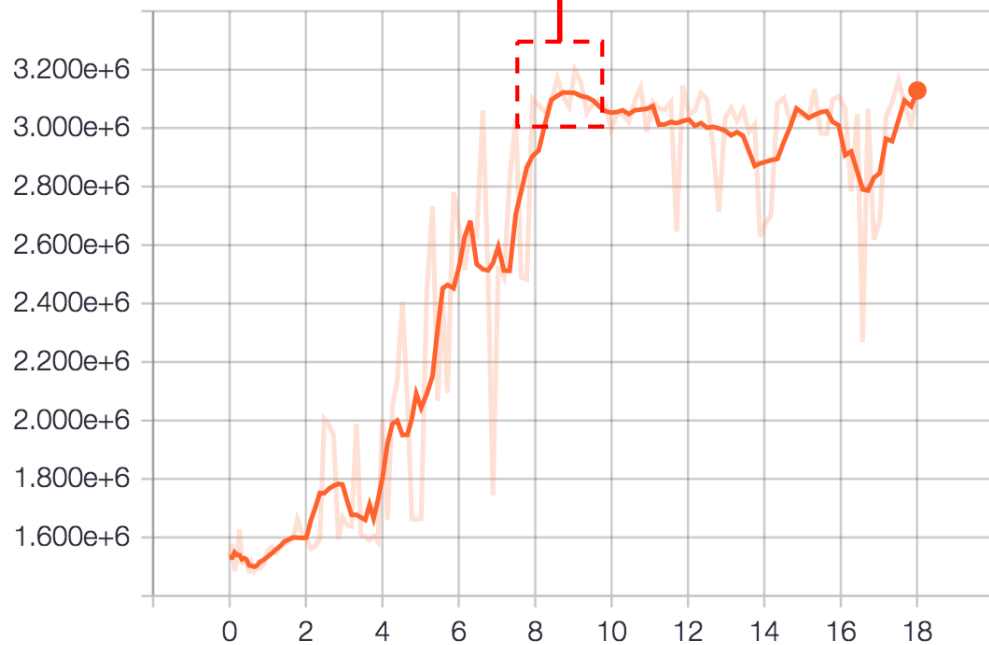
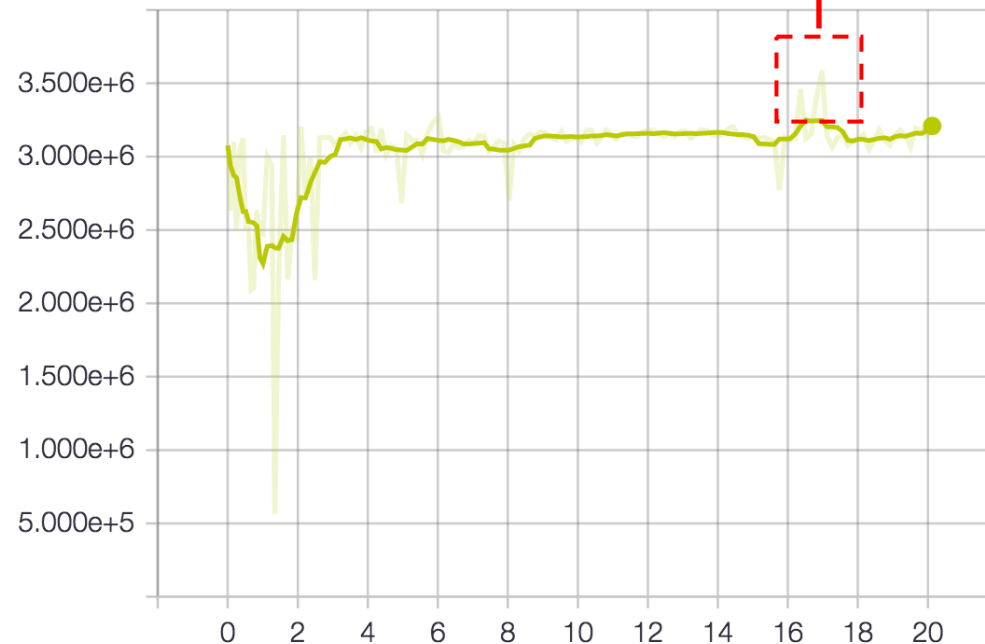
하나의 모델을 처음부터 학습하기 위해선

"반복된 실험의 학습 시간을

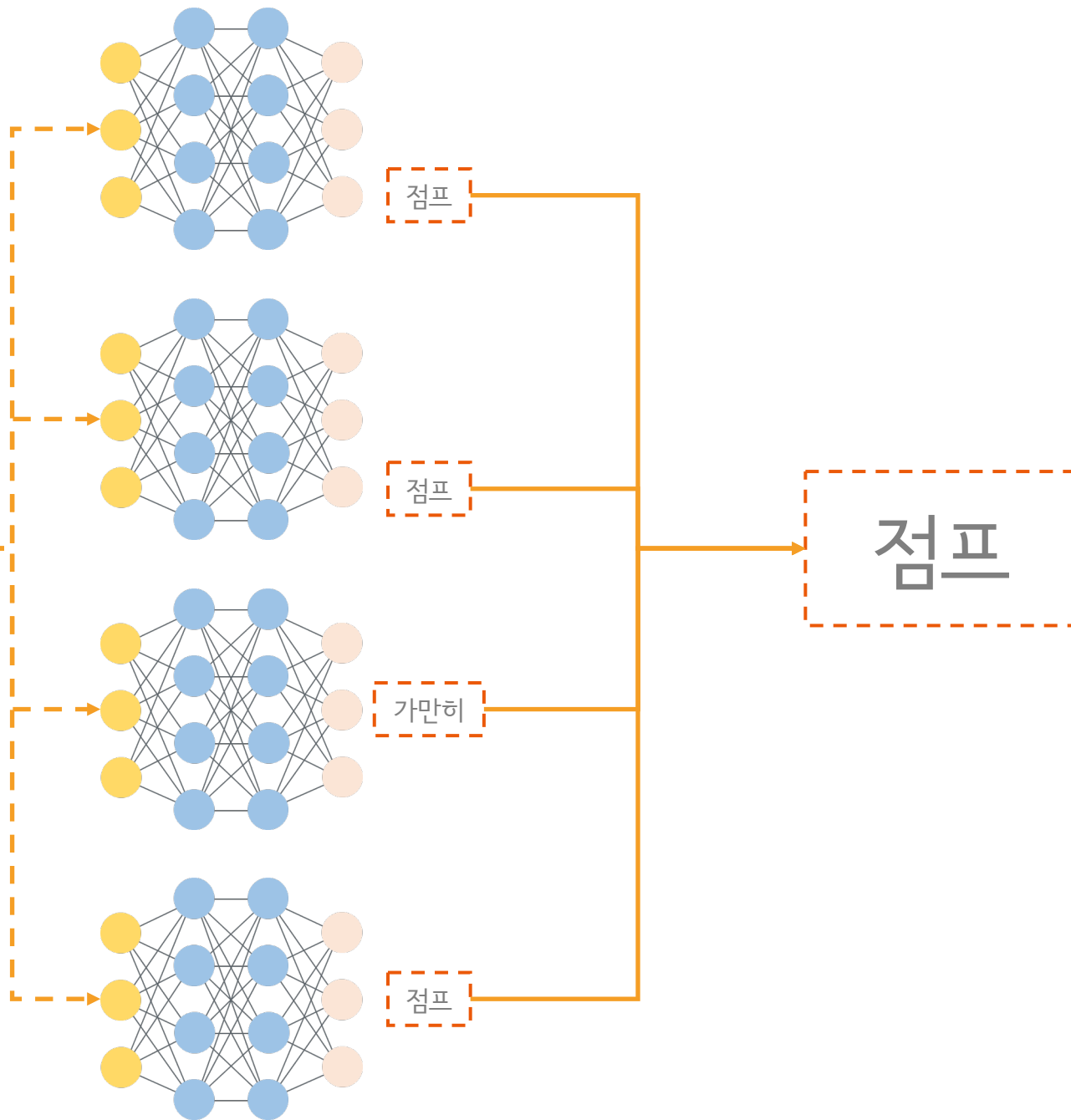
단축시켜야 한다."

모든 네트워크의 weight를 저장하고,

새로운 실험을 할 때

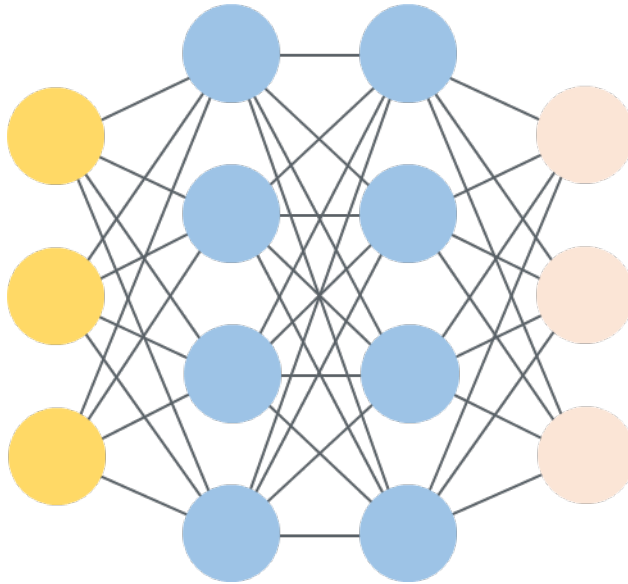
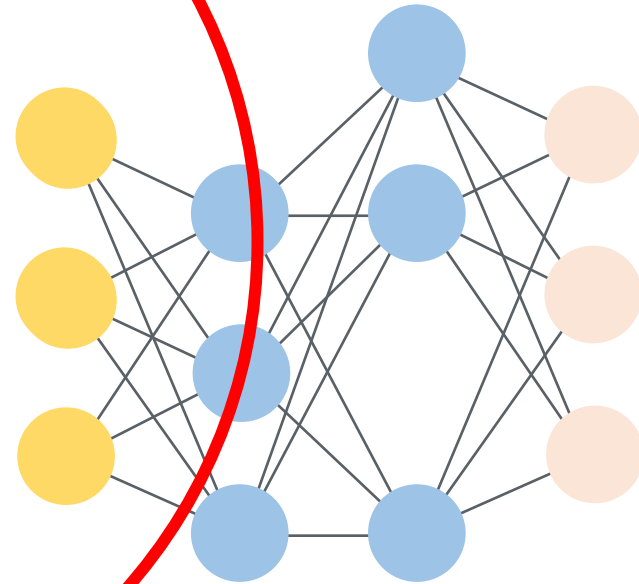
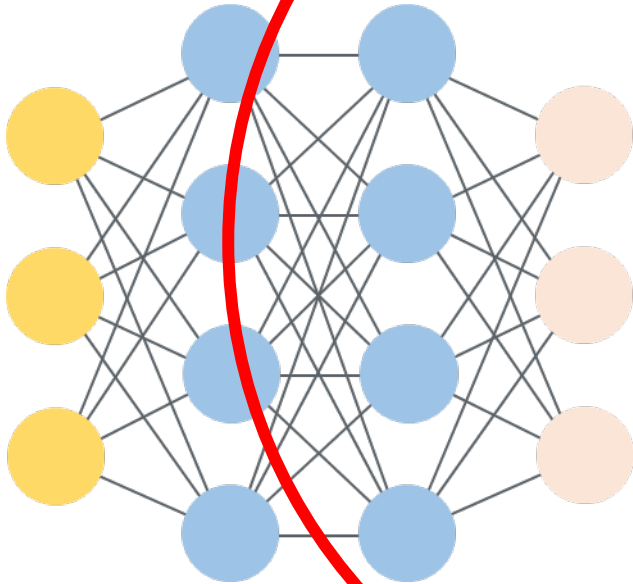
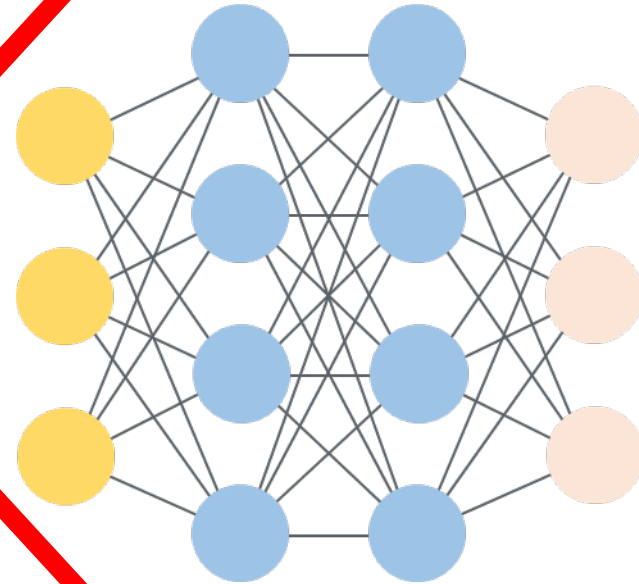
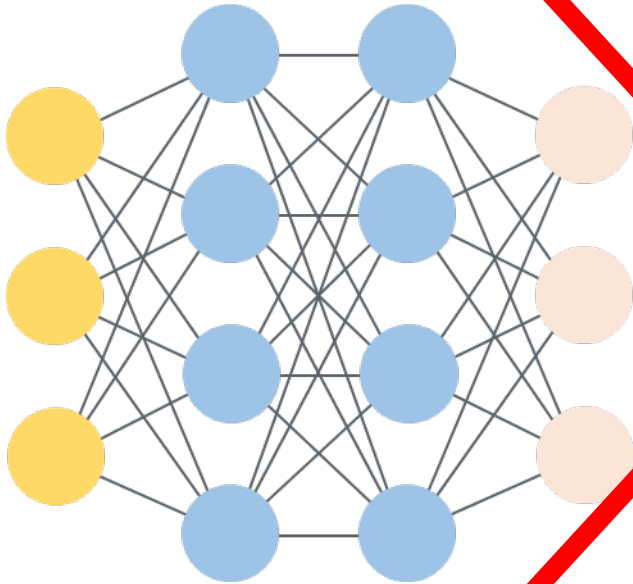
비슷한 실험의 weight를 처음부터 사용

더 높은 점수를 얻을 확률이 높다

4. Ensemble methods

Experiment #1
Experiment #2
Experiment #3
Experiment #4
S
점프
점프
가만히
점프
점프

“하나의 실험에서 만들어진
여러 weight들을
동시에 로드 하려면?”

Ex) 가장 잘했던 weight는
보너스 타임은 잘하는데,
두번째로 잘하는 weight는
젤리를 잘 먹는다

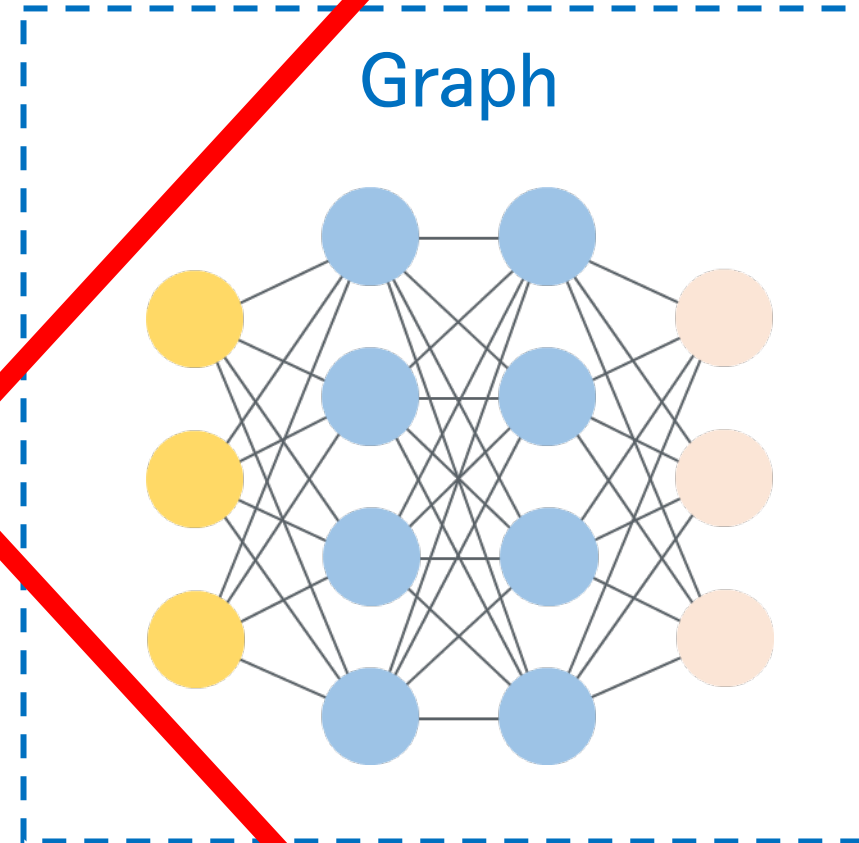
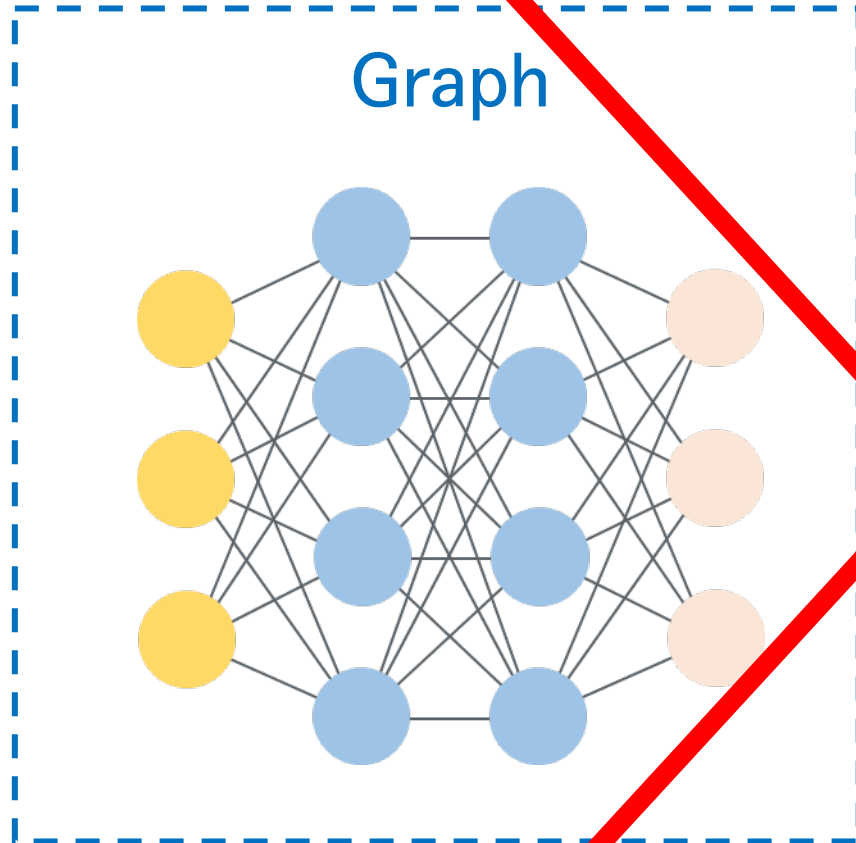
Session

Session
Graph

Session
Graph

Session
Graph

Session
Graph

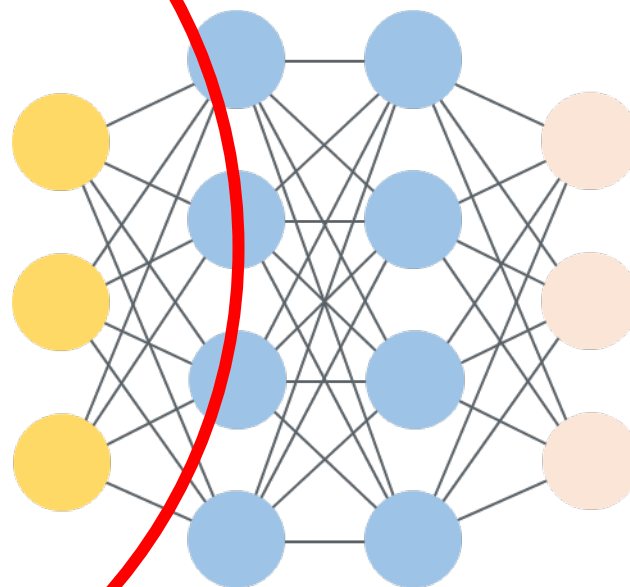
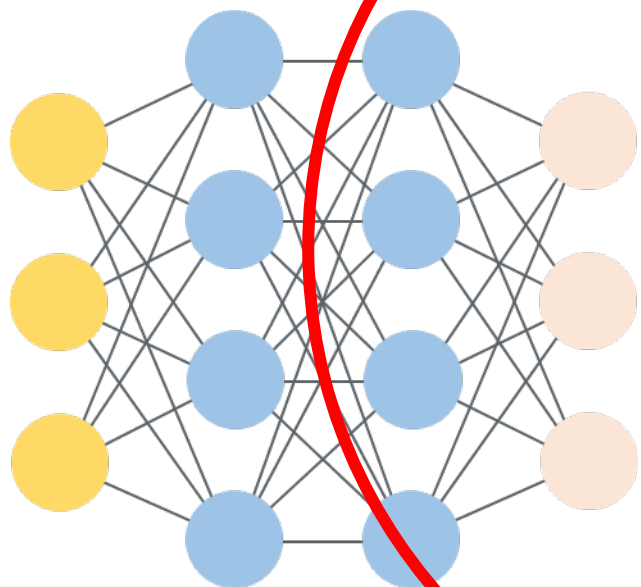
같은 이름의 node는

하나의 그래프에

두개 이상 존재할 수 없다

Session
Graph
Graph

한 세션에는

하나의 그래프만 존재

Session
Session
Graph
Graph

핵심은,

평소처럼 이렇게 선언하지 마시고

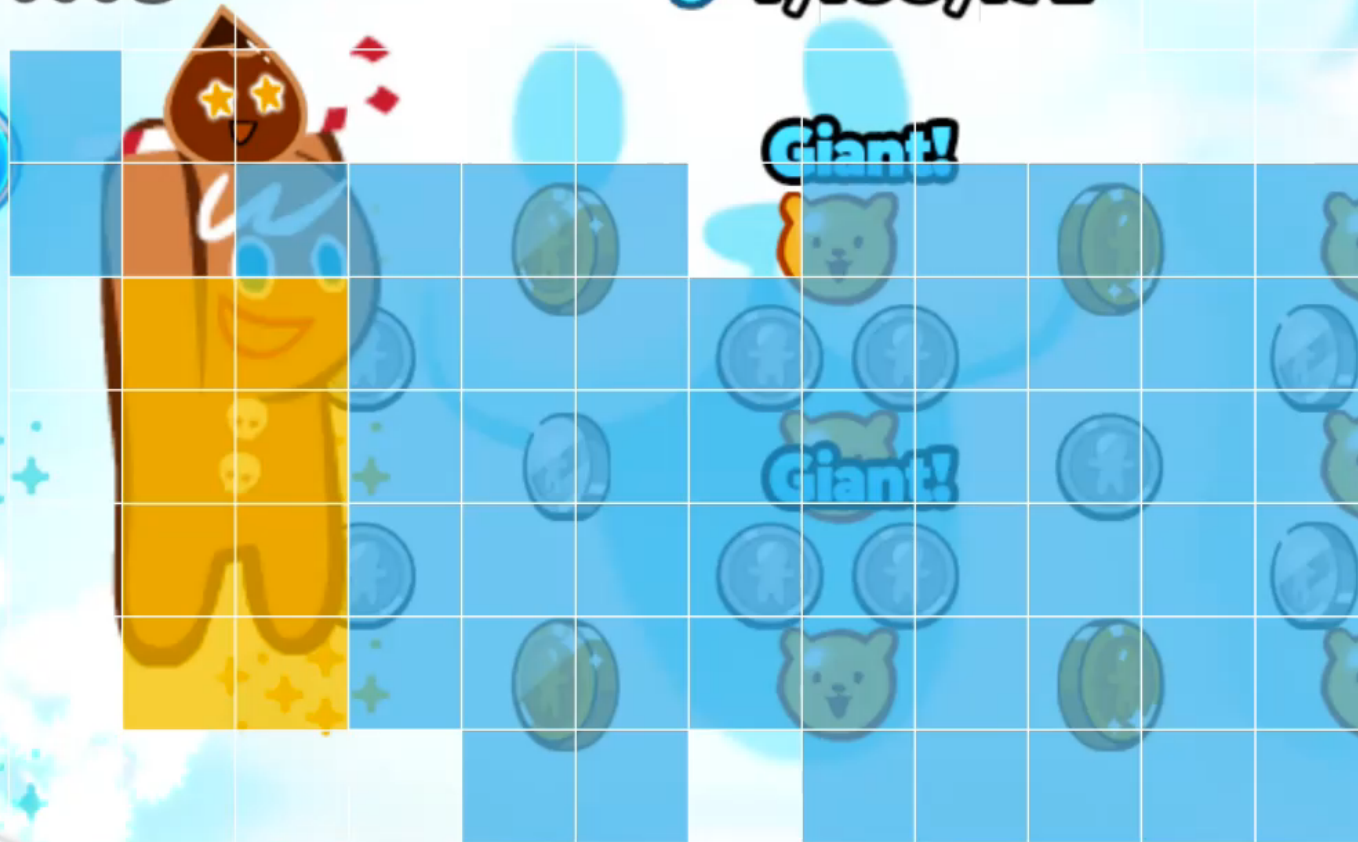
```
with tf.session() as sess:
    network = Network()
    sess.run(network.output)
```

이렇게 Session을 살려서 선언하시면 됩니다

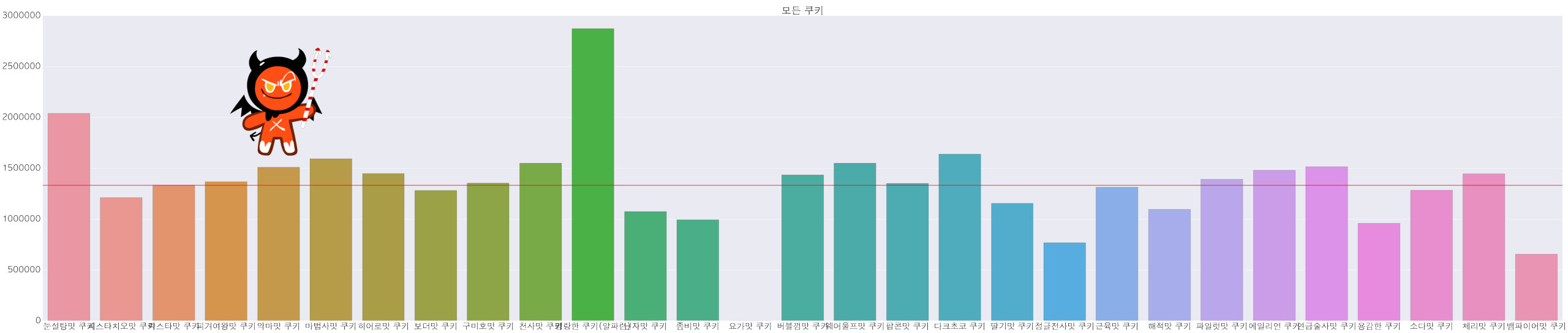
```
sessions = []
g = tf.Graph()
with g.as_default():
    network = Network()
  sess = tf.Session(graph=g)
  sessions.append(sess)

sessions[0].run(network.output)
```

같은 방식으로

보너스 타임도 학습!

BONUSTIME
1,188,172
651
221/400
1
Giant!
Giant!
Jump
Slide
GL verts: 2436
GL calls: 26

쿠키런 A.I.로 밸런싱 자동화 하기

밸런스를 360배 빠르게 해 봅시다

DEVSISTERS

학습된 A.I.로 모든 쿠키의 평균적인 점수를 계산하거나

소다맛 쿠키

2000000
1500000
1000000
500000
0

털뭉치 멍뭉이 럭키다이스 마법사전 스페이스 미니볼 로켓폭죽 야생 고기왕 젤리코 큐브 낄낄 폭탄 브레인껌 반딧불이 생명 저울 초코 방울 여우구슬 샌드형 스피커 구름 펠리칸 불꽃박쥐 꼬마 유령 스포트라이트 치즈 방울 조각레몬 천상의 별 영혼 투구 미니 잭슨 2호 참나무 주스통 눈꽃송이 쌍둥이 덤벨 까칠한 토마토 스노우글로브 포켓 딸기

펫을 바꿔 보면서 성능 차이를 확인하거나

네,

“알파런 잘 뜁니다.”

감사합니다